Complexity Reduction of CNNs using Multi-Scale Group Convolution for IoT Edge Sensors

Qingyuan Wang, Antoine Frappé, Benoit Larras, Barry Cardiff, Deepu John

Complexity Reduction of CNNs using Multi-Scale Group Convolution for IoT Edge Sensors

Qingyuan Wang*, Antoine Frappé†, Benoit Larras†, Barry Cardiff‡, Deepu John‡
*‡University College Dublin, Ireland
†Univ. Lille, CNRS, Centrale Lille, Junia, Univ. Polytechnique Hauts-de-France, UMR 8520-IEMN, France
*qingyuan.wang@ucdconnect.ie, †{antoine.frappe, benoit.larras}@junia.com, ‡{barry.cardiff, deepu.john}@ucd.ie

***Abstract*—In this paper, we propose Multi-Scale Group Convolution (MSGC) an optimization to the conventional convolutional layer, to address the high computational complexity issue in deploying convolutional neural networks (CNN) on the Internet of Things (IoT) enabled edge sensors. The proposed method reduces complexity by grouping input channels of a convolution layer into smaller groups, thereby reducing the number of intermediate connections and complexity of matrix computations in a CNN. This approach results in a minor performance loss, which is compensated by utilizing a characteristic of group convolution to extract multi-scale features. The proposed technique is applied for detecting cardiac arrhythmias from electrocardiogram (ECG) data using CNNs to be deployed in edge sensors. For the binary classification of ECG into Normal or Anomalous beats, the proposed MSGC-based CNN achieved an average 30% reduction in computations while achieving similar or better performance compared to the conventional CNNs. We used the Physionet MIT-BIH Arrhythmia database for performance evaluation, and in the best scenario, our approach increases accuracy by 0.47%, F1 score by 1.87% while only using 64.41% MACs and 83.62% parameters. This optimization strategy can be extended to other CNN models where computational complexity reduction is critical for deployment in edge devices.**

I. Introduction

Cardiovascular diseases (CVDs) are one of the leading healthcare challenges in terms of risks, costs, and mortality [1]. The symptoms of CVDs can be sporadic, sudden, and can go unnoticed. Therefore continuous monitoring and a fast diagnosis are needed for CVD patients even outside the hospital premises. Deploying machine learning (ML) techniques on wearable sensors is widely considered a promising approach for continuous cardiac arrhythmia monitoring and detection. On-device edge intelligence using deep neural networks (DNNs) can enable long-term continuous health monitoring and early warnings so that proactive actions can be taken without causing privacy concerns.

Convolutional neural networks (CNN) are commonly used for CVD monitoring and detection [2] due to their high accuracies. CNN's inbuilt signal filtering and shift invariance feature make it more robust than perceptron networks. CNNs are also friendly to parallel computation and thus hardware implementations. Many studies employed 1D-CNN approaches with good detection performance [3, 4]. Several 2D-CNN based techniques that process images generated from transform domain features of ECG signals are also reported [5, 6]. All these works claim overall detection accuracy of above 90%. However, their models are usually too large and require enormous computational resources, preventing them from efficiently deploying in low-cost, low-power embedded IoT sensors. It is possible to reduce the model complexity by directly reducing the model size. Nevertheless, this may result in a significant performance drop and defeat the purpose.

This publication has emanated from research supported in part by 1) a grant from Science Foundation Ireland under Grant number 18/CRT/6183 and 2) CHIST-ERA grant JEDAI CHIST-ERA-18-ACAI-003.

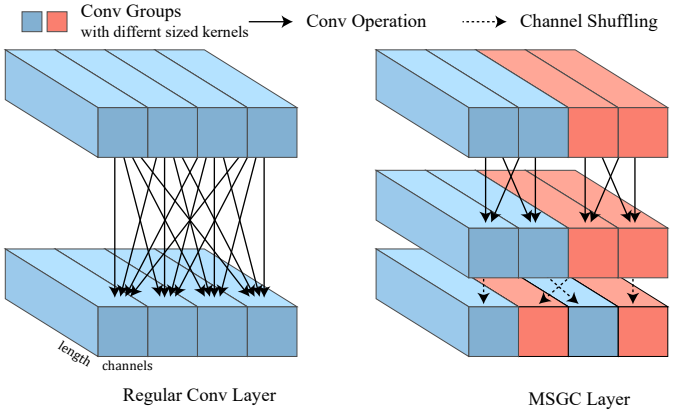


Fig. 1: A Typical CNN layer vs an MSGC layer

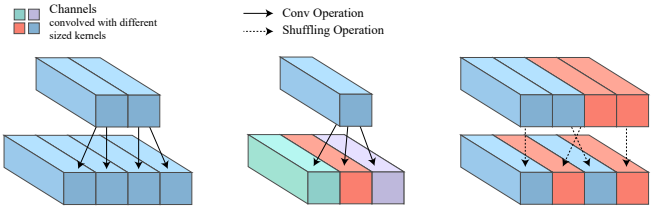
Model compression techniques are often used for the complexity reduction of DNNs so that they can be deployed on resource-limited devices. Common techniques include 1) Model pruning, 2) Quantization and 3) Knowledge distillation etc. Model pruning removes redundant and less significant weights from a trained neural network, reducing computations and model parameters. Quantization replaces trained weights with fewer-bit representations to limit the complexity of arithmetic operations. Knowledge distillation helps to train a simplified (smaller) model using the original (larger) model so that the smaller model can replace the larger one [7]. These methods sacrifice performance for efficiency and can be applied universally to any model. Domain-specific knowledge also could be used for model compression. For instance, MobileNet [8] is an optimized CNN architecture designed for imaging tasks on mobile devices. Neural architecture search is a technique that automatically searches for the best architecture [9]. These techniques use prior knowledge or simulation results to optimize the model and do not necessarily reduce performance.

In this paper, we propose an architecture optimization technique for CNNs and apply it for binary classification of ECG for CVD event detection. We present Multi-Scale Group Convolution (MSGC), an optimization for the convolutional layer in CNNs, for reducing the computational complexity for deployment in resource-limited devices.

The difference between a typical Convolutional and MSGC layer is illustrated in Fig.1. MSGC is a combination of group convolution [10, 11], multi-scale feature extraction [12], and channel shuffling [13]. Group convolution reduces convolutional layer complexity by grouping input channels into smaller groups, thereby reducing the number of intermediate connections and complexity of matrix computations. This results in a minor performance loss, which is then compensated by the latter two techniques. While reducing complexity and model size, the proposed technique maintains a similar detection performance to that of a conventional CNN-based model. This technique is inspired by CNN optimization methods used in computer vision. The effect of this method is evaluated using the Physionet MIT-BIH Arrhythmia database. The details of MSGC are presented in Sec.II. Experimental details, Complexity comparisons with conventional CNNs, and Results are discussed in Sec.III.

II. Multi-Scale Group Convolution

The primary motivation of MSGC is to minimize computations while retaining the ability for feature extraction from the input. As Fig.2 shows, MSGC combines 1) Group convolution, 2) Multi-scale feature extraction, and 3) Channel shuffling into a single layer to achieve this. Group convolution reduces computation by eliminating connections between different groups. This also allows for different kernel (filter) sizes in different groups, encouraging the model to view the input one different scales. After the convolutions, channel shuffling (a computation-free operation) is done to exchange information among multiple groups and create combinations of different kernel sizes with more layers stacking together.



(a) Group Convolution. Each group has two separate output channels

(b) Multi-Scale Feature Extraction. One channel is convolved with three different sized kernels.

(c) Channel Shuffling. Two groups are swapping their channels.

Fig. 2: Components of Multi-Scale Group Convolution

A. Group Convolution

In group convolution, the input channels are divided into smaller groups, and convolution operation is done within each group. Fig.2a shows an example of separated input channels, each of which has two independent output channels. This method was initially proposed with AlexNet [10] to distribute the model on two GPUs. While, the main advantage in our application is that it can reduce the convolution layer complexity and may not result in a performance drop [11]. This is because group convolution works like an internal micro ensemble of multiple independent submodels, and this can compensate for losses compared to a single large model. However, this advantage requires that every submodel has enough ability to make an independent prediction. For very small models, the submodels from group convolution are less effective in making independent predictions, and therefore it may not improve and instead possibly reduce the performance. The following methods are used to compensate for any performance loss.

B. Multi-Scale Feature Extraction

The characteristics of useful features could vary slightly in every input sample. For ECG, the signal morphology can vary 1) temporally due to user activity, 2) measurement methods, 3) anatomical variations etc. These variations in signal characteristics make feature extraction less effective if we only use fixed kernels. DNNs are able to address this issue if the model size is big enough and with a sufficiently large training dataset. However, it is difficult for a size-limited model to achieve this by training only. Artificially introducing more prior knowledge, also known as applying inductive bias, is beneficial for feature extraction in this case. In our proposed method, we assign different kernel sizes for convolution groups, as this will enable the model to view the input signal on different scales. There are other successful precedents reported for this technique. In Rocket [12], a large number of untrained convolution kernels were used to cover feature extraction from all possible input variations. This could be considered an extreme case of pre-defined multi-scale feature extraction. In our model, we assigned different kernel sizes to different groups of output channels to enable multi-scale feature extraction. Note that these multi-scale groups are not necessarily the same as the groups in the group convolution. This means different kernel sizes can be used even when the group convolution is not applied. Fig.2b illustrates a case of one input channel being convolved in three different ways.

C. Channel Shuffling

The sparse channel connections due to group convolution operation make every group independent. However, after stacking layers, each channel will never communicate with other channels outside its group as they are not connected. One complexity-friendly method to establish communication between channels is shuffling them after group convolution, which was first employed in ShuffleNet [13]. The shuffling operation can be implemented by re-indexing the channels and does not involve any computation. Furthermore, channel shuffling will create a mixture of various kernel sizes. In our implementation, channel shuffling is designed to make all combinations among groups. Hence, the shuffling is fixed and the number of channels must be the multiple of g^2, where g is the number of groups. Fig.2c shows a shuffling between four channels of two groups.

III. Experiments

The MSGC-based model is evaluated by comparing it with a typical CNN model. We built several low-complexity networks based on the architecture in Fig.3. We denote the size of input signal as $[l_{in}, c_{in}]$. The network will change the channel numbers to c_h while decreasing the length gradually. At the end of these models, a small fully-connected network with n_h neurons will make the binary prediction. The complexity of these models are estimated in the number of multiply–accumulate (MAC) operation and number of parameters while their performances are evaluated in accuracy (ACC) and F1 score [2].

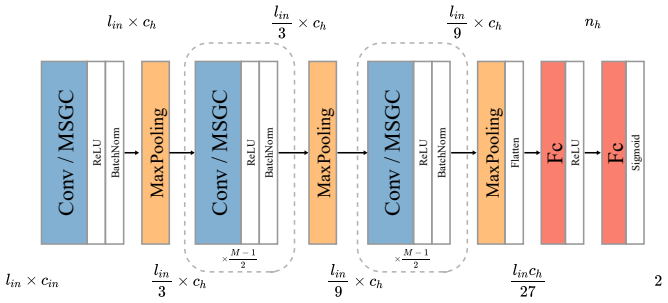


Fig. 3: Model Architecture used in our experiments

A. Dataset and Data Pre-processing

We used the MIT-BIH Arrhythmia database for performance evaluation. It contains 48 half-hours, two lead ECG records, sampled at 360 Hz. We used a single lead (MLII) signal as the input and extracted 240 sample points centered at the R-peak location as a beat sample. The signal was normalized to follow the standard normal distribution to reduce any impact of amplitude variations across multiple records. The beat annotations are converted into two classes, i.e. normal and abnormal, to support binary classification. We divided the dataset into DS1 and DS2 as per the recommendation of the Association for the Advancement of Medical Instrumentation (AAMI) [14]. The records in DS1 are used for global training. From DS2, the first 5 minutes of each record is used for patient-specific training, and the rest are used for testing as per [14]. After the pre-processing steps, our experiments involve a total of 93,391 beats, about 11% of them are abnormal beats. ∼47k (50%), ∼7.4k (8%) and ∼3.9k (42%) beats are used for global training, patient-specific training and testing, respectively.

B. Experiment Details

The models are implemented with PyTorch, and their complexity is estimated using ptflops [15]. All models are firstly trained with the global training set for 200 epochs. Then, for every record in DS2, the global model will be further trained using the first 5 minutes of data for 50 epochs and is evaluated on the rest of the record. The patient-specific model will be reset to the global trained state after evaluating each DS2 record. We applied some common training boosters such as an advanced optimizer, learning rate scheduler and data augmentation method to improve performance and stability during both training phases. AdamW [16] is used as the optimizer with a learning rate of 0.0003, and a cosine annealing scheduler is used to adjust the learning rate. Mixup [17] is also applied to reduce the over-fitting issue.

Our experiments include six cases of different tiny models. The experiment starts from a model with 3 Conv layers and two fully-connected layers, as illustrated in Fig 3. It has 9 hidden channels (denoted by c_h) for all Conv layers, and 64 hidden neurons (denoted by n_h) for fully-connected layers, i.e. $c_h = 9$ and $n_h = 64$. The purpose of the experiment is to validate if MSGC can be a replacement for conventional convolution in our application. We also tested some network variations with more channels and layers to evaluate MSGC's generalisability. Specifically, the width of networks can be extended to 18 hidden layers and 128 hidden neurons, i.e. $c_h = 18$ and $n_h = 128$. The number of Conv layers (denoted by M) can also be increased to 5 or 7.

We also tested the components of MSGC separately to verify our hypothesis that the three methods can work together. Specifically, we tested four types of convolution: 1) group convolution alone, 2) MSGC without channel shuffling, 3) complete MSGC, and 4) conventional convolution. Deep learning algorithms are not deterministic and may sometimes be unstable when the model size and training dataset are small. Therefore, we repeated all experiments with different seeds three times to increase the results' reliability. The metrics in Tab.I are the average of those three repeats.

C. Results

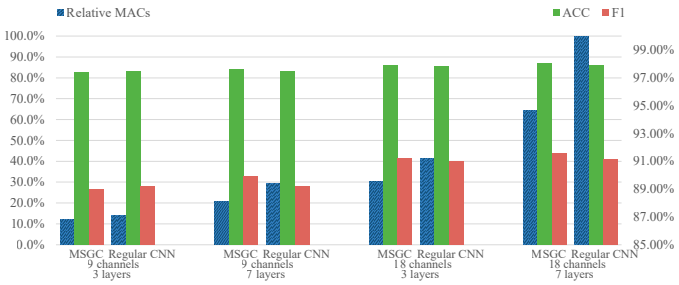


Fig. 4: Results comparison between regular CNN and MSGC

Fig. 4 visualizes four test cases comparing between MSGC and conventional convolution. The blue bar shows the relative MACs between displayed cases. A considerable saving of MACs is observed in every case. The green and red bars represent the accuracy and F1 score. The performance is nearly unchanged between the two convolution types. Meanwhile, we can see a trend of more complexity results in better F1 score. Tab.I compares the simulation results of six test cases. The proposed method achieved a significant reduction in MAC operations for all test cases. Increasing the number of convolution layers does not result in superior results but increasing the number of hidden channels improves performance. Regarding different convolution types, using group convolution only has the lowest complexity. It reduces the

complexity significantly with a slight drop in performance. The MSGC layer can achieve similar or even better performances to the regular CNN layer with reduced complexities. In the best case ($M = 5,\ c_h = 9,\ n_h = 64$), 26.11% of MACs and 7.05% of parameters are removed while the accuracy and F1 score are improved by 0.50% and 1.75%. Applying MSGC with channel shuffling improves the ACC and F1 scores in almost all cases, with no increase in complexity. However, we note one exception for the network with $M = 7$, $c_h = 18$ and $n_h = 128$. In this case, MSGC without channel shuffling outperforms the complete MSGC, achieving an accuracy of 98.41% and an F1 score of 93.01%. We think this is because this configuration coincidentally suits the dataset. The performance improvement of MSGC is from the multi-scale operation that captures features on different scales. In practice, enabling or disabling channel shuffling, varying the number of Conv layers and hidden channels will make the model sensitive to different scales.

Channels	Conv Layers	Conv Type	MACs	Parameters	ACC	F1
9	7	Regular Conv	111,757	6,411	97.51%	89.21%
		MSGC	78,853 (70.56%)	5,781 (90.17%)	97.65% (+0.14%)	89.94% (+0.73%)
		MSGC No CS*	78,853 (70.56%)	5,781 (90.17%)	97.03% (-0.49%)	87.03% (-2.19%)
		Group Conv	57,361 (51.33%)	5,439 (84.84%)	97.33% (-0.19%)	88.32% (-0.90%)
	5	Regular Conv	82,183	5,871	97.22%	88.31%
		MSGC	60,727 (73.89%)	5,457 (92.95%)	97.72% (+0.50%)	90.06% (+1.75%)
		MSGC No CS	60,727 (73.89%)	5,457 (92.95%)	97.63% (+0.41%)	89.77% (+1.46%)
		Group Conv	44,959 (54.71%)	5,223 (88.96%)	97.28% (+0.06%)	87.92% (-0.39%)
	3	Regular Conv	52,609	5,331	97.50%	89.20%
		MSGC	45,481 (86.45%)	5,133 (96.29%)	97.45% (-0.05%)	89.00% (-0.20%)
		MSGC No CS	45,481 (86.45%)	5,133 (96.29%)	97.45% (-0.05%)	88.87% (-0.32%)
		Group Conv	35,437 (67.36%)	5,007 (93.92%)	97.15% (-0.35%)	87.63% (-1.56%)
18	7	Regular Conv	377,869	15,609	97.94%	91.14%
		MSGC	243,373 (64.41%)	13,053 (83.62%)	98.06% (+0.12%)	91.61% (+0.47%)
		MSGC No CS	243,373 (64.41%)	13,053 (83.62%)	98.41% (+0.47%)	93.01% (+1.87%)
		Group Conv	166,045 (43.94%)	11,721 (75.09%)	97.85% (-0.09%)	90.66% (-0.48%)
	5	Regular Conv	267,205	13,557	98.01%	91.45%
		MSGC	178,501 (66.80%)	11,865 (87.52%)	98.01% (-0.01%)	91.43% (-0.02%)
		MSGC No CS	178,501 (66.80%)	11,865 (87.52%)	97.95% (-0.07%)	91.19% (-0.25%)
		Group Conv	124,069 (46.43%)	10,965 (80.88%)	97.93% (-0.08%)	90.99% (-0.46%)
	3	Regular Conv	156,541	11,505	97.87%	90.99%
		MSGC	113,629 (72.59%)	10,677 (92.80%)	97.95% (+0.08%)	91.26% (+0.26%)
		MSGC No CS	113,629 (72.59%)	10,677 (92.80%)	97.80% (-0.07%)	90.59% (-0.41%)
		Group Conv	82,093 (52.44%)	10,209 (88.74%)	98.01% (+0.14%)	91.39% (+0.40%)

* MSGC No CS refers to Multi-Scale Group Convolution without Channel Shuffling.

TABLE I: Simulation Results of model variants using different convolutional layers.

IV. Conclusion

This paper proposes a novel optimization to the conventional convolution layer which can be an efficient replacement for convolutional layers in low complexity models. The experimental results show that the MSGC version of the network consistently outperformed the original version of the network. Furthermore, this improvement in complexity is achieved without sacrificing performance, and this model optimization method can co-exist with other approaches for further model compression. It provides a candidate component for building an AI-based system for low-power, resource-limited edge devices.

References

[1] W. H. Organization, *The World Health Report 2002: Reducing Risks, Promoting Healthy Life*. World Health Organization, 2002.

[2] Z. Ebrahimi, M. Loni, M. Daneshtalab, and A. Gharehbaghi, "A review on deep learning methods for ECG arrhythmia classification," *Expert Systems with Applications: X*, vol. 7, p. 100033, Sep. 2020.

[3] S. L. Oh, E. Y. K. Ng, R. S. Tan, and U. R. Acharya, "Automated diagnosis of arrhythmia using combination of CNN and LSTM techniques with variable length heart beats," *Computers in Biology and Medicine*, vol. 102, pp. 278–287, Nov. 2018.

[4] O. Yildirim, P. Plawiak, R.-S. Tan, and U. R. Acharya, "Arrhythmia detection using deep convolutional neural network with long duration ECG signals," *Computers in Biology and Medicine*, vol. 102, pp. 411–420, Nov. 2018.

[5] "ECG arrhythmia classification using a 2-D convolutional neural network," Apr. 2018, arXiv:1804.06812 [cs].

[6] M. Salem, S. Taheri, and J. Yuan, "ECG Arrhythmia Classification Using Transfer Learning from 2- Dimensional Deep CNN Features," in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Oct. 2018, pp. 1–4, iSSN: 2163-4025.

[7] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A Survey of Model Compression and Acceleration for Deep Neural Networks," *arXiv:1710.09282 [cs]*, Jun. 2020, arXiv: 1710.09282.

[8] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861 [cs]*, 2017.

[9] "Neural Architecture Search: A Survey," Apr. 2019, arXiv:1808.05377 [cs, stat].

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012.

[11] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5987–5995, iSSN: 1063-6919.

[12] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, Sep. 2020, arXiv:1910.13051 [cs, stat].

[13] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pp. 6848–6856.

[14] A.-A. EC57, A. f. t. A. o. M. Instrumentation, and others, "Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms," *Association for the Advancement of Medical Instrumentation, Arlington, VA*, 1998.

[15] V. Sovrasov, "Flops counter for convolutional networks in pytorch framework." [Online]. Available: https://github.com/sovrasov/flops-counter.pytorch

[16] "Decoupled Weight Decay Regularization," Jan. 2019, arXiv:1711.05101 [cs, math].

[17] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," Feb. 2018.